Open Source Week – Day 4

IBM Code Tech Talks

03/01/2018

https://developer.ibm.com/code/techtalks/open-source-week/

>> TODD MOORE: Okay. Welcome everybody to Day 4 of Open Source day here at IBM. My name is Todd Moore. I work with members of the industry to develop new organizations, as well as manage the folks that typically participate full time in many Open Source activities throughout the world.

Today we have a number of talks the first talk is from Fred Reiss, Chief Architect at the IBM Spark Technology Center, the co-author of System ML and prolific inventor and author.

Fred is going to be talking to us today about Apache Spark and hopefully give us some great insights into what's happening in Spark, so Fred I'm going to hand it over to you and we'll get started

>> FREDERICK REISS: All right. Thank you, Todd. I'm going to share my screen here, and fire up my presentation. Can everyone see my presentation out in the audience?

>> We can see it.

>> FREDERICK REISS: Excellent. So as Todd mentioned, I'm Fred Reiss, the Chief Architect here at IBM's Spark Technology Center. Today I'll give a brief introduction to Apache Spark and to the Spark community.

So here is how the presentation is going to go. We're going to start with a brief overview of what is Apache Spark and then we'll move on to talk about the origins of the Spark project and the Open Source Community that has grown up around that project.

After that we'll move into a technical overview of Spark internal, drilling down through the different components of the system. And we'll pop back up and then talk about the overall status of the project and then we'll move on and discuss what's coming next for Apache Spark.

Finally, I'll talk about how IBM is both using and contributing to Apache Spark, and we'll finish up with some helpful resources if you want to learn more about the system., so let's get started.

So, what is Apache Spark in a nutshell? Spark at its core is a system for analyzing huge amounts of data on a cluster of commodity servers using parallel processing. There are three main

reasons that you would use Apache Spark as opposed to other systems in this area, three main benefits of Spark, and those are ease of use, performance, and extensibility.

These three characteristics have made Spark the go-to choice for a wide variety of different analytics tasks, whether you're doing ETL or decision support, machine learning or graph analysis, streaming event detection, whatever analytics task you're looking at, Apache Spark allows you to quickly and easily spin up scalable, high-performance applications using Spark's built-in APIs and rich extensibility.

So the story of Apache Spark goes back to around 2008 at the University of California in Berkeley. Within the Berkeley Computer Science Department, there was a group of data scientists, system builders, and algorithm designers who were known as the AMP Lab. Stands for Algorithms, Machine, and People. One of the students in the lab was Matei, who came from Waterloo in Canada and spent the last few weeks grappling with making machine learning work with massive data with hardware with his Berkeley work. The key challenge is feeding data into the CPU running the algorithm fast enough to keep the CPUs busy.

One way around the problem was to cache the data in main memory across the cluster. While this approach works great on high-performance computing hardware, it was pretty much a non-starter with the inexpensive, unreliable commodity hardware that he had available to him, where a single failure of any node across the entire cluster was enough to bring down the entire application.

So one day I'm told while he was out skiing at an AMP Lab retreat, Matei had an idea. Node failures occur offer enough that you cannot rely on the node failures not happening. But at the same time, the failures are actually infrequent enough in practice that you can afford to spend a few seconds or even a few minutes recovering from one.

So instead of having a really overengineered solution, why not just keep just enough metadata around on the primary node to recompute the data that you lost when you lose a node. That way if the worker node fails, your system can just pause for a moment, reconstruct the missing data, and continue on without a problem.

And this concept came to be known as an RDD, and which stands for Resilient Distributed Dataset. If you have this metadata stored in the Resilient Distributed Dataset you can execute a logical execution plan for a series of computations that lets you build up a scheduler that you can use to choreograph the entire execution and it can happen where the developer doesn’t want to worry about where the node runs or where it might fail, and can you kind of see where this is going.

Here is now in particular the RDD concept worked in the

original Spark system. When an application uses this RDD paradigm, the application logic is going to live entirely in a process on that cluster's master node, and this process is called the driver in Spark terminology.

The driver is at the left of the diagram on the screen right now. When we're taking the application developer out of the business of choreographing what code runs in the other parts of the cluster, the application lives entirely on the driver. All of those decisions about parallelism and redundancy, replication, those are taken care of behind the scenes by two components called the scheduler, which choreographs that shifting and executing the code and a block manager, which keeps track of what partitions are cached in memory where on the cluster.

So the application code, which is running on the driver is going to interact with objects called RDDs. RDD, as I said stands for Resilient Distributed Dataset, and each RDD object represents the partitions result of the parallel computation. That is the RDD has pointers to other objects which in turn contain the logical definitions of the different partitions in this dataset.

Again, these definitions are entirely logical. They don't say anything about whether that computation has even happened or where that computation happened, if it did happen, or where the results are currently residing if they are being cached anywhere.

The only thing that's stored inside this RDD object inside that -- inside that chain of objects I should say, is the logical description, and this description can be recursive, a given partition may be referred to other partitions that infer to other partitions and so on to define an entire data parallel execution plan.

So all of this plan forms a -- oops, sorry. This forms a directed acyclic graph, and you can see you can build out the schedule for performing the execution, and that's exactly what the scheduler component of Spark does.

Now, it traverses the graph and determines what parallel execution steps need to occur on the workers, which are called the executers, in order for the physical contents of a given partition to appear on one of the executers.

If a cached copy of that data is already available, the scheduler can just skip those execution steps and return the cached copy, and if that cached partition disappears at a later time, the scheduler has all the data it needs to schedule new tasks to reconstruct that missing data.

The scheduler needs to spin out the replacement executor to recover from that shift over the code that it has on the driver and redo the computation.

Now, compared with previous approaches, like Hadoop or MPI, the RDD paradigm makes it easier to build scalable applications

because all the applicationable logic is going to reside on the master node. The RDD mechanism hides nearly all of the complexities and failure modes of parallel processing and network communications, sorting and caching, and all of these complexities are hidden from the application developer because the application developer only works on the application running on the driver.

Now, once Matei put together this initial prototype, he shopped it around to different students and professors in the Berkeley AMP lab and they latched on to this idea and helped to build up a full system around the concept, a system which they named Spark.

Now, because RDD rely so heavily on building out the logical descriptions of chains of future computations, the team decided it would be best to use a functional language, in this case, Scala to build the decision. Spark became the most complex code ever implemented in Scala because this project grew and grew and grew.

As other students joined the project they started building libraries on top of the original RDD code. First, there was the machine learning algorithms that were -- that the RDDs were originally designed for, and then because you need some data to feed those machine learning algorithms, other students developed a SQL processing engine on top of RDDs, and then because SQL databases often store graphs, another student built a graph system called GraphX, again on top of RDDs, and then because sometimes these graphs and data stream into the system, another student built a streaming engine on top of RDD that deals in data in what are called microbatches.

And then because all of this needs to run on some infrastructure, the team started building out -- building out the Mesos Data Center, added facilities for managing pools of Spark executers on top of Mesos and eventually it was ported to other resource managers like NARN and Kubernetes.

And now the students working on Alluxio gave data to be able to go outside of the JV heap, it's JVM based.

The system became quite mature and was starting to be used in preproduction and even production by some industrial users, so UC Berkeley decided to denote the code base to the Apache Software Foundation which turned Spark into Apache Spark, and this project started out on the Apache Incubator but very quickly graduated to become a top-level Apache Open Source project in 2014 and since then has moved on to become the most active Apache Open Source project.

Now, if you look at the community that's grown since then, first off, the center of that community you can find at Spark.apache.org. Now the code for itself for Apache Spark is available on Apache's GitHub Repository and it's run out of the Apache JIRA system and there are some also active mailing lists for developers and users, and you can find the mailing lists at the URL I have here on the screen. All of these links, by the way, will

be on the last slide and in the PDF that you can download.

As for conferences, the main conference is Spark Summit soon to be named to Spark and AI Summit that you can find at Sparksummit.org and that's a lot of deep technical content and as well as practical information on how to deploy Spark in the enterprise.

Of course, a very large number of companies, including my employer IBM, are actively contributing to the Open Source project and actively leveraging Spark in their products and services.

Now, the original Spark system, as I mentioned, started out with just RDDs. Now, Apache Spark on the other hand, the larger open project is a much beggar and complex system with many different components, but you can break down all of this complexity in terms of an overall layout that's focused on those three main benefits of Spark that I talked about earlier in the talk, performance, ease of use, and extensibility.

And there is kind of a major chunk of Spark dedicated to each of these attributes, so the ease of use of Apache Spark stems mostly from the user-friendly APIs which started out with RDD and have built up extensively from there.

The systems performance on the other hand is centered around the high-level optimizer for SQL processing and a lot of high performance low-level code in the core of the system, and the extensibility comes from two pluggable APIs at the top and bottom layers of the system that allow Spark to interact freely with third-party libraries, with data sources, and cluster management frameworks such as Yarn and Mesos and Kubernetes.

Let's take a close look at each of these aspects and the components of Spark that drive the aspects in greater detail.

We'll start out with the ease of use layer. So Spark's ease of use stems from this very powerful but very high-level application programming interface or APIs. Now, all of these APIs are very carefully designed around the original core principle of RDD, which is to make large-scale data processing feel like small-scale data processing.

Spark provides all the familiar tools that a data scientist would use if she was analyzing data on her laptop, but it provides all of those tools in the context of a cluster computing framework.

The API starts out by giving full access to Spark's capabilities from the high-level data science-oriented language just like R and Python and then add data frames and stream processing and interactive command lines for data exploration tasks.

Now with the original RDD concept, the application's core logic run, still runs entirely inside the Spark driver. All of that messy parallel processing and tolerance is carefully hidden away without compromising the system's ability to express complex algorithms and without compromising performance, of course.

The layer that has to deal with performance, speaking of performance, is immediately below that API layer, and I like to call this the core of Spark. Spark's core data processing engine, it drives performance with a mixture of brute force and finesse. So the brute force part comes from the system's massive scalability as well as the ability to schedule tasks very quickly and to cache data in memory, and read data very quickly back from memory.

Now, in production use, Spark is in production with thousands of nodes and many users and is executing complex computations every day with low latency on petabytes of main memory data.

Now in addition to this brute force component, there is also a significant amount of finesse that goes into Apache Spark, and that finesse part comes from an advanced SQL optimizer for high-level query processing and lower level graph execution framework that provides lazy evaluation and automatic management of cache information. Spark's data components are known as the Tungsten use dynamic code generation, compression, (?) memory, to make Spark's JVM run time match the performance in many cases exceeded the performance written in other languages like C++.

Now, the third major benefit of Apache Spark is extensibility, and the layers that provide that third benefit are plug in above and below the core and API layers.

At the top of the Spark stack there are numerous libraries that interface with Spark, in addition to the data frame library built into Spark's API layers, Spark also ships with ML Live a collection of scalable machine learning algorithms and GraphX a collection of routines for graph processing. There are literally dozens of analytics third-party built on top of Spark that provide a capability of analytics capabilities from natural language processing all the way down to deep learning.

Now the RDD concept and extension into structured data processing via Spark data frames allows all of these libraries to work together because every parallel execution in Spark is expressed logically in terms of interoperable high-level APIs is very easy to compose multiple Spark-enabled libraries to create a complex and heterogeneous series of computations.

At the bottom layer of Spark, there is a second set of plug-in points for interfacing with other systems that manage data at rest and resource allocation. Data in the enterprise can be used in many different place, can reside in many different place, and it comes in many different formats. Spark's flexible data source API allows spark to access data in a wide variety of distributed file systems and scaled-up database systems. All you need is the right plug-in for your particular data source and there are literally dozens of different plug-ins available.

Likewise, Spark works with multiple different resource managers, Yarn and Mesos are fully supported out of the box in the

version of Spark that was released this week, comes with experimental Kubernetes support, IBM's Spectrum Conductor with Spark product is massive scale enterprise private Cloud deployment, and the Spark Kubernetes Project contains a more bleeding-edge version of the functionality for Kubernetes as being slowly folded into Spark mainline.

Now, Spark also ships with its own standalone resource management for light-weight deployments that can run on top of a collection of virtual machines or bare medal servers. You have as a user of Spark, a wide variety and different choices for deploying your Spark cluster.

Now taken together, all three of these pluggable APIs allow Spark to service the core data processing engine for many different analytics applications, regardless of what storage system, data formats, cluster manager, or kinds of modalities of machine learning the application requires.

So let's tie everything back together here. Now, as I mentioned Apache's key -- Apache Spark's key advantages are ease of use, performance, and its extensibility and each of these characteristics is centered into a single part of Apache Spark's stack so the ease of use comes from a straightforward and powerful extensible interface of the API layer.

Performance stems from the many sophisticated components down deep in Spark's core which uses a combination of finesse and brute force to make all the analytics expressed with API go fast, and the extensibility shows up in the many high-level libraries you can plug into Spark's APIs and as well as the connectors to data sources and cluster computing frameworks that plug in down in Spark's connectors layer.

So now we've taken a brief tour of the internals of Apache Spark, and let's continue with the story of the Apache Spark project, starting where we kind of let off where Spark had become Apache Spark and started to accumulate a wide community.

So the next chapter of the story is happening right now. Today Apache Spark is a vibrant and active project. It's, in fact, one of the most active projects under the Apache Banner. Since becoming, an Apache project in 2014, Spark has put out 16 different releases, the most recent of which was just this week, and has included code contributions from over a thousand members of the Open Source Community.

Now the core project is managed by a group of about 50 committers; although, that's about to grow closer to 60 once the next set of committers goes through. All of these committers are Open Source developers granted the permission to deliver code changes directly to Apache Spark, and they work for a wide variety of different companies. Top employers of Spark committers include data bricks, IBM, Yahoo and all of these committers interact with

each other, developers, review code, and work with the hundreds of other active contributors to the project to keep the project moving forward.

Now, the user community of Apache Spark is equally vibrant. Over 25,000 questions tagged Apache-spark on Stack overflow and 10,000 threads on the Spark mailing list. Over 3,000 unique users of Spark's Project Management System on Apache JIRA, and in general, Spark is really the mainstay of the analytics infrastructure at many enterprises worldwide and there are many high-profile examples of production use.

Now, as the Spark project moves forward into the future, we can expect to see a lot of continued improvement and continued hardening of the core data processing capabilities of the system as well as new libraries to provide additional machine learning capabilities both in Spark's built-in ML library and in third-party libraries.

Spark's streaming performance continues to improve and some experimental features in Spark 2.3 are moving that forward by leaps and bounds, and the capabilities of that Spark on Kubernetes project are being folded into Apache Spark to provide native Kubernetes support with an experimental version of that being released just this week, and also deep learning support is coming into Spark via several different parallel projects.

So Apache Spark is very important to us here at IBM. IBM is a major contributor to the Spark project through the IBM Spark Technology Center where I work. We've made ourselves over 900 contributions to Spark. We have lots of active work going on in areas including SQL, Machine Learning, Python, APIs. We also package and distribute Apache Spark itself. I should say IBM as a whole, packages and distributes Apache Spark itself through several different vectors both on the IBM Cloud as well as in our private Cloud offerings, and Spark helps drive the analytics capabilities internally in over 25 different IBM products and that number keeps going up every year.

I've included some links at the bottom of this slide if you'd like to find out more about IBM's commitment to Apache Spark. The first URL here will take you to our general information page on Spark-based product lines and the second link, Spark.tc is the homepage for the Spark technology center. The third link is the link to Apache's project management facility, JIRA where you can -- rather to a dashboard on top of that facility that will show you an active list, a live list of IBM contributions to Apache Spark. We also have pages down at the bottom here that show links to other projects from IBM in the general Spark ecosystem.

So here are some additional resources if you'd like to get started with using Apache Spark for your own analytics project or learning more about the system, so you can download the latest

version, the latest Open Source version of Apache Spark to your laptop from the Downloads Page on Spark.apache.org. If you want to manage parallel Spark on the Cloud, take a look at the IBM data science which provides, Can you Log In and Start Up a Spark Cluster Instantly.

We also provide many Spark-based Jupyter notebooks that give good examples of how to do many different data science tasks using Apache Spark. If you want to look at official documentation for reference purposes you can find that off of our website, and if you want to see more in-depth learning material take a look at the Apache Spark portions at cognitiveclasp ai. Or look at the examples on developer.IBM.com in the code patterns that show you how you solve some very interesting end-to-end data science business-oriented problems using Apache Spark.

There are lots of different ways to become a member of the Apache Spark Community. You can, of course, become a user of Spark or you can get involved directly in the project. Can you join mailing lists, you can contribute to code, you can attend one of many different local meetups in your area to learn about the latest Spark news, conferences such as Spark Summit and Strata. If you're an enterprise user, consider the IBM Spark Advisory Council that is the non-commercial forum that promotes the adoption of Apache Spark across different enterprises and helps to steer the Spark Technology Center code contributions to better meet the needs of enterprise users.

And of course, you can try out Apache Spark for free on the IBM Cloud and just go to IBM.com/Cloud/Spark.

Thank you very much for attending my presentation. I hope you have enjoyed this brief introduction to Apache Spark. Once again, I'm Fred Reiss, Chief Architect at the IBM Spark Technology Center. Wishing you a great day and hoping that you have a wonderful rest of the week.

So at this time, if there are any questions from the audience, I'd be happy to take questions for the remainder of my time slot.

>> MODERATOR: There is just one question, how does Spark ML compare or relate to Tensor Flow or ML using R?

>> FREDERICK REISS: The question is how does Spark ML or in particular Spark's built-in ML compare with Tensorflow or with the capabilities you would find in R?

>> Yes.

>> FREDERICK REISS: So these different systems are largely complementary and a lot of users of Spark, myself included, use all three of those at the same time, so Spark's ML primitives are built from the ground up for high scalability. So if you need to train, for example, a logistic regression model over very large and sparse data that doesn't fit on one machine, Spark ML is the way to go for that.

The ML capabilities that come from Tensorflow are geared more toward deep learning application, so if you have a problem in your data science pipeline that has some very noisy feature space, maybe involves some images or text, you probably want to use deep learning for at least part of that problem and for that you want to have a deep learning system.

There are several different libraries that interface Tensorflow with Apache Spark, including Tensorframes, Tensorflow on Spark and Google is working on making Spark a producer of streaming data to feed into Tensorflow Distributed.

As far as more traditional frameworks like R and the stack that run on a single machine, those frameworks have been around for a lot longer than Apache Spark, and they have implemented this long tail of functionally, so chances are if you need a more obscure algorithms, you can find an implementation of an algorithm and as long as your data fits on one machine, you can run it with R or Python, but because Apache Spark integrates with these frameworks, and in particular Apache Spark has native APIs for both R and Python, you can combine these libraries, these single-machine libraries, with parallel execution for the portions of your application that require that parallel execution.

So just to sum it up, all three of these machine learning libraries are parts of a data scientist's toolkit, and you would use different tools for different task, but they all work very well together. Thanks for that great question, by the way.

>> Thank you for the great answer, Fred. If there are no more questions, we will take a short break here and get set up for our next session. Just take a quick pause and we'll be right back. Thank you.

>> FREDERICK REISS: Thank you.

>> TODD MOORE: Okay. Are we ready to continue?

>> Sure, go ahead.

>> TODD MOORE: Okay. So if you're just joining us, this is Todd Moore. IBM's Vice-President for Open Source Technology. I'm responsible for the work that we do route in Open Source. Many of IBM's developers who spend their time working out in the many communities around the world work for me, but I also spend quite a bit of time out working in communities helping them become established and promoting them and helping where I can in Open Source.

In this session I'm going to introduce two folks, Susan Malaika. Susan is the Senior Technical Staff member in the IBM Digital Business Group, and a member of IBM's Academy of Technology. And within our group, she's responsible for all data-related technologies. She's been, you know, tirelessly working to increase adoption and contribution to Open Source, and then engages in quite a few different communities, including the ODPI Consortium and of

course many AI members of ecosystem and the Janice Graph Project that is out at the Linux Foundation. So you can find Susan pretty much anywhere around the world holding a hackathon or meetup or being engaged in universities. She does quite a bit out in the Middle East as well for us, and she'll be one of the speakers here today along with Luciano Resende.

Now, Luciano is one of our Data Science Platform Architects here at the Spark Technology Center at IBM, and he contributes to many Open Source projects in the Apache Software Foundation. He's been there over 10 years working mostly on things contributing to Big Data-related Apache projects and of course the Apache Spark Ecosystem, so he's going to also be talking to us because he's part of Jupyter, and between Susan and Luciano, I think we'll have an interesting talk here for you.

Let me hand it over to you, Susan, and you can do further introductions to get us going.

>> SUSAN MALAIKA: Thank you very much, Todd. It's over to you, Luciano. Luciano, can you share your screen?

>> LUCIANO RESENDE: Thank you. Thank you, Todd. Thank you, Susan. Let me share my screen here. Let's get started.

So good morning and good afternoon and good evening. Today we're going to be talking and giving you an update on the Jupyter Community. So a quick look at the agenda, we're going to be talking about what is Jupyter, what are Jupyter notebooks, how the project was created, how the community is going, and go over some technical overview, talk about current status and options of Jupyter not only in IBM but like in the community in general, and some calls to action.

So, without further ado, let's get started. So for the folks that are not familiar with Jupyter, Jupyter kind of brings a -- it's a web interface. It will bring interactive development for the public in general. It gets very famous or very adopted on the data science community because it enables you to combine the code execution, rich text, mathematics, plots, and rich media. You can have the code that is doing some logic or an algorithm, and surrounding that you can have the explanation of like what that piece of code is doing. You can pull out and have organization on results of that code that helps you get a better understanding of the data, a better understanding of like the results that you are getting, and you can start sharing those and collaborating with different data scientists so that demonstrates to be a very powerful thing among the community and is now being widely used in different areas of the industry.

In terms of like scenarios that it's being used, data claims, ETLs, transformations in general, numeric simulation, statistics, modeling, machine learning, AI, and much more. You can see a link here for more information of like use cases.

On the right side, you see two kind of like UIs here. We're

going to be talking more about the details of that, but I just want to show that one on the top is just a Jupyter Notebook, as is kind of more like a single application, a single notebook for an entire UI. The community has been working on a Jupyter Lab which is kind of the next generation, and you can see that there is going to more like an ID and provides you like tabs and ability to switch between projects, have cooperations between each notebooks and that's the direction that the project is taking.

And notebooks, in general, have become so popular, they now even have like their own conference. JupyterCON was held first time last year in New York City sponsored by O'Reilly, and we'll have JupyterCON around the same timeframe, if I remember correctly, in New York. And because it's so popular, they're starting to have things more communicated where they have having now what they call a Jupyter pop-up and there is going to be one on the week of 20 -- I think it's 21st of March in Boston, and I think they have plans to have another one as well before the end of the year.

Still, what is Jupyter? So, this is a little bit more details on how things are. You see that you have a server that serves the notebooks or the browser and what I want to mention here is that mostly those notebooks doesn't go -- they're not tied specifically to one technology. They have what we call Kronos that allows you to abstract it and you can pull different ones and they will support different languages, Python, Scala, R, you name it, there are over a hundred existing kernels.

And this is a little more what people have been talking about. Jupyter, it's a new from data (?). Some of the terminology, Jupyter, Project Jupyter, Jupyter Notesbook. JupyterHub is more multi-user and JupyterLab is like the next-generation UI that the community is taking about.

So, how was Jupyter created? In the past people have basically shells tossed some of the development, so there was the Python shell, and they wanted to come up with a little bit more richer in terms of a UI for that, and that's how like Fernando Perez initially created the iPython Environment Notebook and that started providing a web interface so you can run broadly and get some of those computations in a more -- in a better kind of like user experience for the developers.

That got very popular and at some point, the iPython became kind of just like the support for Python, and we started seeing much more proliferating which is kind of complicating the Jupyter Project.

As a nation, JupyterLab is becoming kind of like the new UI more towards an ID for the data science development, it's still based on notebooks. The kernels in the back are mostly the same. Mostly what they're doing is transitioning the UI to make it more user friendly, more with application development.

And up until a couple of weeks ago, it was mostly like just development getting to use it and heavy developer. But on February 20, they have announced that JupyterLab is basically ready for general users, and so we encourage people to go and come and try and provide feedback and things like that so that we can get it like really GA soon.

For more information, there is a link here for the JupyterLab announcement. There are details how you install and everything. You can see on the right Fernando Perez, the creator and a couple other co-founders and contributors into the Jupyter, Sylvain, Brian, and Jason, you can see a lot of folks coming from the academia world and they use a lot of that in academia for simulations and for even using that for teaching students and being able to have like interactive development sometimes necessarily for teaching.

So where is the Jupyter community? If we look into GitHub, the Jupyter community is in four GitHub organizations. We have the main one, which is Jupyter, and that's where the actual notebook project is hosted and if we look into that, we have today about 323 contributors and over 10K commits there.

JupyterHub is one of the main projects as well. It kind of has a mission to distribution and it basically will spawn Jupyter Notebooks and different nodes for a given user that has around 88 contributors, and around 2,500 commits.

JupyterLab, which is kind of the next version of Jupyter, it's around the UI side as well, so 106 contributors with 11,680 commits as of this morning.

Also, projects that are kind of innovating in one of the part of the Jupyter Ecosystem. There is what we call a Jupyter Incubator. Currently there is -- there are six active projects at the Jupyter Incubator and kind of like around all of those six projects, we have counted like 56 contributors and around 1,700 commits as of this morning.

And any other information and things like that you can see at Jupyter.org, the URL at the bottom here.

The whole community mass, what is called a Jupyter Steering Committee and those are members of the committee that has been contributing for a while, they are very active, and they help kind of like manage the community and make sure that those projects are held and people are taking care of contributing to those. And you can see here some of the participants, not necessarily in any order.

So let's jump a little bit into technical overview. When we look into the internals of the notebook, it has a web server. The web server will serve files and those files are notebooks itself and those are servers and users most likely look to them in the web browsers.

On the right side when you execute code, those are abstract into Kronos and the code is submitted into the Kronos and it will

do the processing and return that back to the client.

And the client, they are kind of like a text editor and also has the capability to do plots and like have documentation and different other syntaxes.

We mention also a little bit about JupyterHub, which is kind of like trying to get Jupyter into a distributed mode. What it does is it kind of provides a multi-user hub which then will have a proxy and that will basically forward each user to a particular notebook that has been spawned for that user that gets kind of like the distribution mode. Also, because Jupyter server itself doesn't provide that capability, so JupyterHub kind of like comes to that to make -- to represent that level and make it multi-user by providing multiple instance of the notebook to users in general.

And another project that is becoming the main project now in JupyterLab, if we look at the left side we can see it's a much richer UI. You'll see that it has multiple tabs, and those tabs can have associated cells. Can you see you can navigate to different projects on the left side? It's really going much more powered like an ID.

There is a link if you want to go more and see like a demo of JupyterLab so also can you install and make sure you get a feeling for that and start working with the kernels and see how it makes better for development and cooperation.

But also, if you want to go into much more like technical deep dive of the architecture, there is a diagram here on the right side where you can get to know more of the internals of JupyterLab and also in terms of documentation available on the website.

In terms of the current status of the Jupyter Community, it's a very vibrant community. We have a lot of over 700 attendees at JupyterCON last year and we're expecting even more and they're going to not only one conference but like multiple, kind of like local conferences, a little bit smaller but local to give more access to other people.

The GitHub organization shown here are very active as well. I've been participating on there mainly and like a year or so ago it used to be like very slow in terms of like maybe a couple of emails every other day, and then we are seeing now much more traffic as well so we see like the community in all aspects are really growing.

So what's next for the Jupyter Community? I think that we have been seeing a lot of like adoption, a lot of like enthusiastic people coming from different areas. We see enterprise adoption, we see adoption in academia, and in other user scenarios.

Basically, we want to continue growing the number of contributors that are, as I mention, the three main projects that people can come and contribute as needed and make new cases, help with the documentation, help with Kronos and things like that. Also, Jupyter Incubator if you have a nice project or utility that

you can be surrounded in some of the notebooks, is usually welcome, and also projects that make (?) better, and I think it's something that we like seeing as part of the community.

As I mentioned, different areas we see it's very spread in terms of like people in research, education, media industry and much more. Always welcoming new contributors and new use cases and new contributions as well.

In terms of like Jupyter at IBM and adoption at IBM, Susan, do you want to go over a little bit on this?

>> SUSAN MALAIKA: Yes, I can do so. So we're finding that Jupyter is growing organically all over IBM, and indeed in the communities at large without us having to promote it in any way because it's just so natural to use.

There is education materials in CognitiveClass, there is a link there. And there are also a lot of journals, or what you'll call patents, you'll find a great many of those in the patents collection on the code part of IBM.

And also, you'll find in the Data Science Community a lot of notebook usage as well. Do you want to go through the next slide?

Now, one of the areas where Jupyter surprisingly became very popular was in the IBM Commuting Group, so we suddenly found that they were using Jupyter notebooks for all of their -- when they document how to access IBM's quantum computers on the Internet, they show how to do that through APIs from within Jupyter Notebook, so that's a very exciting usage of Jupyter.

There is also some external projects such as the Brunel Visualizations that use Jupyter Notebooks.

There is a project called PixieDust that has a lot of libraries to use Jupyter in a greater amount of environment, and of course, the Jupyter Enterprise Gateway which Luciano is going to tell us a lot about. The data Science Experience also has a lot of Jupyter Notebooks. Over to the next slide. Back to you, Luciano.

>> LUCIANO RESENDE: Thank you very much. So Jupyter Enterprise Gateway is one of the projects currently in Jupyter Incubator. It originally started with my team here in IBM, and we contributed to Open Source, so the notion behind enterprise gateway is like when we started seeing -- when we started looking into building an analytics platform, we started seeing a lot of limitations when we wanted to, in terms of like scalability, security, and kind of like a user impersonation. And those are kind of three of the main areas that enterprise gateway kind of like came to help.

It's -- it enables also running and launching the kernels into remote servers that are scaled linearly to the number of clusters that you have, and enables you to support a lot more data scientists coming with notebooks and running those on expensive (?), particularly when you're doing data, deep learning type of

workloads, that is very important.

This project has a mission at the Jupyter Incubator. We're (?) Open Source and we also have been using that in a couple of projects internally that is available to use, for example, for the IBM Data Science Experience that uses the enterprise gateway to connect to a cluster and be able to run all of those kernels remotely.

In terms of like getting started and just coming to try, as Susan mentioned, there are various code patterns available. You can start playing with PixieDust. The Data Science Experience provides you a more integrated environment that brings a lot of like other tools around surrounding the notebooks and also lots of like example notebooks that you can start working and playing with that.

Also, meetups for Python data science and machine learning, we have been participating to those and talking about and general areas about the notebook usage and also going into deep technical aspects of like building analytics platform. We usually participate and have like Jupyter education but also, we participate in the (?) and we're going to be in March in Boston going to the Jupyter pop-up as well.

So getting to try one of the code patterns to participate in the meetup communities close to you, or if you're traveling on business or somehow you find another place that might have one, so it's good to meet new people, do some networking and learn a little bit more. Membership links around Jupyter and also like GitHub locations where you can start contributing. With that, I'm open to questions if there are any.

>> Luciano, there is one question, or two questions, one that Susan already answered but if you like to add some to it, you can. If you would like to add additional features to JupyterHub or Lab, how would you inject your need or request for enhancement to the project?

>> LUCIANO RESENDE: That is a very good question. So these projects are mostly run on GitHub so new issues can be created and if you are bringing up like a very cool new feature. People can start forwarding to that or committing to that which will catch more attention from the community. But also, there is if you go to Jupyter.org, you will see a link for a mailing list that people can also communicate there and the mailing list is usually for if you see a bug, if you have questions, announcements, and that is kind of like a one list for all the Jupyter projects. So it's good because if you have things that are multi-projects or you don't know if it's more on the front-end side or the backend, I have one list and it's easier for you to get information from multiple people and all the active contributors. And even Fernando is still very active there and will respond to your request and questions.

>> The next one, how does Jupyter kernel compare to R studio, and are there any plans to converge?

>> LUCIANO RESENDE: So I think the difference -- I think R Studio it's more like a rich client that will give you a different set of capabilities but I think internally when you look into some of the execution, depending on the dependencies that you have a similar R run time executing those, more like a difference in UI and capabilities. And so plans to converge, I think I'm not aware of plans to converge at the moment. I can take a look at that.

>> Sounds good. The next one is, can Jupyter kernels leverage (?) inside containers?

>> LUCIANO RESENDE: So that is a very good question. So, the way you do this in the community today so the notebook requires the Kronos to be locally, and so the way JupyterHub does is they will launch a container that will have the kernels co-located with the notebook. What Enterprise Gateway is doing, Enterprise Gateway is working on supporting the coupling the kernels and have the kernels actually being running, just the kernel in its own container and then you can have a notebook that is remote, maybe like in your laptop or running as a service connecting to that kernel that is running on the container. We should have that kind of like coming out very soon on the enterprise gateway available for general use.

>> Excellent. Last one. Susan already answered it in the chat room and she gave a link for the comparison between Jupyter kernel and Zeppelin.

>> LUCIANO RESENDE: Okay. The link sounds good and yeah.

>> Yep. That's it.

>> TODD MOORE: Okay. If we have no more questions, I'd like to thank the audience. It was a good session. Thank you, Luciano, for your insights and Susan.

We're going to take a short pause here between sessions to get set up for the next one, and we will be returning shortly for Session Number 3 on Python.

>> LUCIANO RESENDE: Thank you very much.

>> MODERATOR: For those just joining us, my name is Todd Moore, Vice-President of Open Source Technology here for IBM Worldwide. With us today we have several members of the Open Source Community, Susan Malaika, who is one of our members of the IBM Academy of Technology and Senior Technical Staff Member at IBM Digital Business Group.

Susan works with many open technology teams in the industry to promote Open Source and contribution. She's focused on data governance right now at ODPI, but works on many other AI pieces of the ecosystem, and she's one of the founders of the Janice Graph project in the Linux Foundation so you can find Susan out and about talking globally, doing workshops, hackathons, meetups and engaged with universities on a worldwide basis.

Also, today for this topic, we'll have Jeremy Nilmeier. He's a Data Scientist, an Engineer, a Developer here at the Spark

Technology Center in San Francisco. Jeremy works on Apache Spark, you know, development. He does teaching and community outreach as well. Basically, everything all things Spark, and he's -- he has some deep understanding of what we do here. He's one of our prolific authors as well too, as well as an all-around nice guy. We're going to get going here, Susan, if there is anything you want to add, or I'll just introduce --

>> SUSAN MALAIKA: Just go ahead, Jeremy. Thank you very much, Todd.

>> TODD MOORE: Okay.

>> JEREMY NILMEIER: Hello, and thank you for inviting me to speak on the Python and the status of the community. It's an honor to be here.

So, I'll just jump right into the presentation. Today we're just going to give a background of what the Python language is all about, how it was created, where the Python Community exists today, some technical background, give you a sense of what the current status of the Python Project is, what's next, and how Python and IBM have been interacting in recent years, and a call to action and how to get started and how to get involved in the Python Community.

So, Python is becoming a very widely adopted dynamically typed programming language. It's popular in many domains. It really sits pretty much everywhere you would expect to see code, including website creation, Dev Ops, definitely scientific computing as well as data science and machine learning, you'll see it quite a bit.

The Python Software Foundation was founded in 2001, and its mission is to promote and advance the language as most of these Open Source foundations.

So the great thing about Python is it's very, very easy to learn. It runs on virtually every system, including new experimental architectures, which is a big appeal. It has extensive libraries for many applications, data science, artificial intelligence, and graphics processing, et cetera.

So website developers will use it and it's not unusual to see a full stack development process written entirely in Python. You can really touch every piece of a web application that you need to access using a Python API from big data to web frameworks to develop exposing risk APIs, and so the functionality is not limited at all and it's not unusual to see an entire pipeline built in Python.

Data scientists love to use Python Py it is Data platform because it's such an expressive easy to read and maintain language. The intent of it was to look like a pseudocode or algorithmic language rather than just with all of the details of a compiled code, and so very appealing to those who approach code from an algorithmic perspective. AI developers love to use it, and this includes all of the latest and greatest deep learning tools, Keras, Tensorflow, and Py Torch and almost all have a Python API and this is true for

pretty much any type of new hardware that comes out because so much of the code is really sitting on C and so you can have a new GPU architectures and memory architectures and you'll almost always see a Python API provided before you'll see any other API.

So the Python Package Index is where most of the libraries sit, and you can get over 100,000 code packages now, and it's very easy to access, it's just pip install and the name of the package and you're good to go.

So some of the more popular examples are things like NumPy, SciPy, Pandas, and we'll talk about some of these projects, but it is -- it is a very strong language and probably I think probably the most popular language out there at this time.

It was created by Guido Van Rossum, he's now dictator for life, I think is now his official or unofficial title, and it's very readable as well and it exposed a (?) re-evaluate print loop, so in that sense it's a interpreted language which allows you to very rapidly explore all of the syntax and makes for very rapid learn cycle for beginners and you can continue to use that paradigm as you become more a developer.

It's really displacing C and Java as the first language to learn; these compiling languages are still very popular but you can get very far with Python and it's really a great introduction. It's also taught in primary schools, it's now a more popular language for primary school students to learn than French, and so it is increasingly popular among young students and I have personal experience with this. I was teaching Python to my daughter when she was in 5th grade and there were 3rd grade students there learning it and learning it enthusiastically.

So this is a language that you can pick up very early on and it is -- it has a lot of power so this is a language that will grow and it's worth the investment to learn in a very deep way.

In 2017, this became the most popular language to visit on stack overflow as well and now it's just growing exponentially, essentially.

The -- you know, Python is basically a layer on top of C, and so this is all Open Source, it's managed by the non-profit organization called the Python Software Foundation and Guido Van Rossum is President of that foundation.

So where does this community sit? And you know it basically sits everywhere. Enterprise developers, you'll see the folks in industry, and again the full stack application developers will -- you'll see entire shops built around Python and it's not -- in many times even though it's a very rapid prototyping language, it's not the -- you know, it's not the type of language that you have to outgrow or rewrite as long as it acts as efficient APIs in the process, so you can commit to having a large code base in Python.

Data scientists and AI developers love it because of the algorithmic nature of the syntax and it really is a tool of choice here as well as in scientific computing. In academic environments, you'll see Python again not only as the first language but the language that lasts throughout -- throughout, for example, a graduate student's career. They'll start writing their code in Python and the entire code base will be released in Python, so this is -- this has been adopted in the academic community maybe a bit earlier on than the enterprise community and so you also have a good pipeline of university students that are very well versed in Python, so it's a good skill set to also recruit for.

There are some -- here are some of the, you know, Python is better for data manipulation and repeated tasks, and R is good for ad hoc analysis and exploring datasets. A lot of times there is an ecosystem between folks who learn R and learn Python. R is a very good language. It still probably has larger machine learning capabilities than Python, other than perhaps the deep learning API access, but a lot of the machine learning in Python is probably an ancestor of R based on some of the inherited designs such as data frames and things like that. It's very similar, and an R developer can pretty easily adopt their skill set to Python if needed.

And so the Python Community is pretty much everywhere around the world. There are lots and lots of meetups around the world, and as Todd mentioned, Susan is around the world hosting and promoting these meetups and so they are virtually everywhere. If you're interested in attending PyCon those are the big conferences but there are lots of other meetups that you can look to get engaged with your local community. So there are lots of resources there for you if you're interested in getting involved.

So, the PyData Stack that has about 3-million users, the big PyData packages, the ones really widely adopted are NumPy and this library is, you know, written basically a layer on top of a highly optimized C code and (?) code that processes arrays and matrix operations, you know, highly efficient ways, so once you access this API you're accessing some very, very optimized code.

And this allows you to do all kinds of scientific computing and with an easy-to-use API that runs very, very efficiently, so this is really probably the library that really put Python on the map for the scientific and machine learning community. SciPy also has lots of additional functionality for more scientific algorithms and things like four-year transforms and things like that that you see more in the scientific community.

Pandas has the -- basically, the data frame API which is similar to the R Data Frame API and also similar to the Spark Data Frame API. This data frame API kind of feeds into the SciKit Learn library which is a very comprehensive machine learning library. Not quite as comprehensive as perhaps the R Machine Learning

Library, but in slightly more comprehensive than the PySpark Machine Learning Libraries, but it certainly occupies a very nice space in the desktop scale machine learning algorithm libraries, so a great package to learn.

Matplotlab is the go-to plotting library for Python and it has a API very similar to Matlab so people who historically come from engineering disciplines really embraced there Matlab as the default library for plotting, flexible library, creates publication-quality images, and very closely follows what you would expect from Matlab.

Theano, I'm not as familiar with, but has some machine learning capabilities and integrates with M of NumPy I think is very similar. So most of these can be downloaded and can you install them individually. We'll talk about in a little bit there is something called the Anaconda project that will allow you to install of these all at once which -- and handle all of these with the dependencies that can be somewhat complex especially in the case of getting NumPy to work in its optimal state.

And now the focus is also on organization -- NumFOCUS is an organization that sponsors the PY activities and conferences.

So the Conda is a large Open Source project, and there are some premium models that are introduced in various places, but for the most part you can download it and you have free access to a slightly higher level of curation for libraries, so for example the NumPy, in particular that installation is much easier if you download the Conda package but then you get Jupyter and plotting libraries for free, so it's a single installation that provides you with, you know, at least -- I think here we listed over 10 high-quality packages that are really the ones that are really widely used in the field, so to encourage you to use the Anaconda installation if you get a chance, it will give you a lot of stuff for free and you can explore that. It also has a GUI and other things like that that make it really fun -- it's a real fun entry point into Python.

So this kind of co-exists with "pip install" as well so if you want to install something directly with pip you can, and if you want to install something that sits in the Conda installation, you just do "Conda Install" and you're good to go either way.

So what's the current status of Python? We've got lots and lots of SDKs, we've got the Anaconda curated data science package for Python, and then there is lots more here on the Python packages.

So the latest Python release I believe is 3.6. Historically, so even on newer machines, you'll see 2.7 as the default installation or version -- some version of 2.7. There was a big hiccup between Version 2 and Version 3, but I think now for the most part people will adopt -- so there were some API-breaking changes between Python 3 and Python 2, and now people are pretty much over the hump now

and Python 3 is now much more widely adopted so download the latest and greatest and enjoy.

So what's next? I think, you know, definitely more growth, data and AI developers, academia. Again, in is a language that you will see, you know, the latest and greatest hardware, the latest and greatest software will almost invariably have a Python API. People increasingly recognize that even low-level GP accelerated libraries will get much wider adoption if it has a Python API, so you will see Python APIs in most cases long before you'll see Java APIs, for example.

So here is a great quote from Stack Overflow, you know, there were the rise of machine learning, it's not a surprise to me, there were no Python programmers in my company except for me. Now all we're use something Python, and I've seen it before in groups and it just catches on, and it's just really so fast. Large scale persistent projects including those with complex graphical users interfaces will have -- will be developed in Python over long periods of time, so you'll definitely -- you'll definitely see it a lot in practice.

I think maybe I wanted to let Susan go ahead and discuss some of the tool interconnects and courses in maybe the next few slides. Susan, are you there?

>> SUSAN MALAIKA: I am here. Yes. Yes. I see there is a question in the chat there about Python compatibility and --

>> JEREMY NILMEIER: Oh, okay.

>> SUSAN MALAIKA: Yeah, you may want to answer it. Go ahead, Jeremy, answer it and then I'll proceed.

>> JEREMY NILMEIER: Okay. Let me find the chat.

>> SUSAN MALAIKA: It's okay. Just talk about compatibility of Python.

>> JEREMY NILMEIER: Okay.

>> SUSAN MALAIKA: No need to get rid of the slides. There we go. Yeah. Just talk a little bit about that with compatibility. In essence it's the major releases that are not compatible and the minor releases are, but go ahead, Jeremy.

>> JEREMY NILMEIER: Yeah. I mean, there was a big hiccup between Python 2 and Python 3 just because there were a lot of API-breaking changes in that version and the option was very -- the option of 3s with a bit sluggish but I think now it's definitely much more widely adopted. It's definitely a better API, it just took a while to migrate the code over. And there were massive API-breaking changes, but you know just enough to cause significant rewrites.

So but now -- but now it's fine. You can probably -- if you were to begin -- certainly, if you're writing scripts and learning you can probably write it in such a way that both APIs will be able to run, so I hope that answers that question.

>> SUSAN MALAIKA: That's great. Thank you. Thank you, Jeremy. Yeah. So there are so many Python materials because the topic is so well established that it almost seems redundant to even mention what's available and so on, but there are many, many projects that are Python based and indeed, Jupyter that we just talked about earlier in the series is of course a Python project, and there are many Python, IBM Data Science Experience and in the code community.

You'll find also in IBM's CognitiveClass there are a lot of Python courses that can you get started to learn Python, and as well as at the developer works, the IBM Developer Works site and there is a quick start on the Cloud as well that's the final link on Page 11 which shows you how to get started with Python and the Cloud.

One of the distinguished engineers that loves Python in IBM is Jean-Francois Puget, can always get his attention if you talk about Python and machine learning, and David Taieb is another distinguished engineer very active in the Python arena.

Moving to the next slide, so yes. Again, there are amazing O'Reilly books that will tell you about Python, there are community organizations all over the world, and we mentioned that I travel a lot earlier, and for example I was speaking in a town in Saudi Arabia, not even the capital, and I was speaking to a group of about 100 students and most were business studies rather than even data science and I asked for a show of hands who knew Python, and these were mainly business studies, and a third were programming and Python, so the Python movement is not just U.S. based, it's everywhere. As well as the CognitiveClass and the Python books, there is the amazing Python conferences, PyData and which are run by NumFOCUS organizations and they're international and the PyCon have days that they on board you to be able to contribute to Open Source projects and I highly recommend, if that's something you'd like to do, to participate in those community days at the Python Project.

JupyterCon also does the same thing and it's just a very good experience because you get the project founders and the project contributors actually helping you get started with contributions and working on issues and so on and producing pull requests.

There are various libraries where you can -- huge numbers of packages like the SciPy and Anaconda also comes with a curated stack and really good packages around data science as well as Python in general.

Moving on to the next slide, so learn about Python at IBM. There is a lot of developer am Works. Join the Python meet-up groups and a lot of the Python meetup groups actually run education, free education themselves and also you can join the NumFOCUS community which is the organization that runs the PyData and PyCon is the developer sprint that I mentioned coming up soon, and that's a very large conference. I think it's in Ohio, or somebody can go and have

a look. As I recall, is it Columbus or somewhere like that?

And then of course there is the IBM Code and Tech Talks mailing lists which of course you can subscribe to.

That's it from the session. Jeremy, did you want to say anything else?

>> JEREMY NILMEIER: No. I think that was all I think we wanted to cover, but I'd be happy to answer any questions. You know, again Python is an exciting project that I personally enjoy writing code and developing algorithms in Python, and glad to be involved with it.

>> SUSAN MALAIKA: And while we're waiting for questions, I will mention that I wrote a blog entry about my experience with Python in a city in Saudi Arabia and it turns out that there are actual Python snakes in demand. I had readers who were reading it thinking it was going to be about snakes. That was a little side effect.

Let's have a look and see if there are any questions.

>> TODD MOORE: There are none.

>> SUSAN MALAIKA: Okay.

>> TODD MOORE: Thank you both, Susan and Jeremy. I appreciate you coming and giving us a little bit more information and lending us your expertise for today.

We are going to move on to our next session in just a few moments, so we're going to take a brief pause here and we will be returning to Open Source Week Day 4. Okay. We'll just be a few more moments here. Kathy, are you set again?

>> Yeah. That's fine. Go whenever you're ready. Just give it a couple minutes in case people are joining for the last one.

>> TODD MOORE: Yeah, we'll give a quick break in between. If you're just joining we finished up the third session for today and we're going to move to Session4 in just a moment or two.

If you're just joining, I'm Todd Moore, IBM Vice-President for Open Technology and I work with the worldwide team doing Open Source development. We're here today to continue into Day 4 of Open Source Week at IBM, and I have the pleasure right now of introducing our session in what's going on in the R Community. Augustina Ragwitz is going to be our speaker. She's a Data Science Advocate for IBM, a long-time veteran in Open Source, has worked in (?) and Pearl and Open Stack and has taken on a role in R where she served on the R consortium marketing committee as well as been organizing the ladies of the R community.

Augustina is going to take us through what's happening in the community and how you too can participate, so Augustina, please take it away.

>> AUGUSTINA RAGWITZ: Great. Can you all hear me okay?

>> TODD MOORE: Yes.

>> AUGUSTINA RAGWITZ: Perfect. Excellent. Hi, so let's

see. So getting started talking about the R Community. R is actually a pretty neat language and a very unique community as we'll discover as I introduce the talk, so today we'll cover what is R, how was R created, where is the R community, a technical overview, as well as the status, what's next, a little bit more about R at IBM and then some additional things on how you can get started with R.

So a little bit about R, it is a free and Open Source language that is used for statistical analysis, and typically students who are exposed to other tools like Matlab and Stata often switch to R because they can run on their own without having to purchase a license, and then R, one of the things I really like about R is that it's been developed for and by researchers who don't come from a traditional software or engineering background, and so some of the workflows for people like myself coming in from a traditional software engineering background are a little bit unusual. It wasn't quite as intuitive for me, but once I understood the workflow and I started using it for research, then it started to make a lot more sense.

So a little bit about R under the hood for those of you that are coming from a more developer background. R is a scheme list inspired implementation of the (?) programming language.

It's a dynamically type language and uses list conventions under the hood for internal representation.

The overall release schedule has an annual XBY.O releases in the spring and patches happen as needed and really, they try to do a final patch release of the previous version shortly before the next big release.

If you want to find out more, there is some links on the slide, so but basically the R-project.org is where you find out some of the things about the R release and R packaging that is structured.

So one of the things that is both awesome and frustrating about R is that it's intended as a tool so doesn't dictate a strict programming style, so much like I think Todd mentioned that I used to do a lot in Pearl, and so (Laughing), and it's interesting the number of R people I meet that actually did used to do a lot of stuff in Pearl and it has a very similar philosophy of there are many ways to do it.

So, it can be adapted to whatever you're comfortable with. If you like OO, you can take a really strict OO approach. If you want to do something more functional programming style, you can do that. If you want to do like a procedural style, it's really about getting things done and less about dictating a particular programming pair paradigm.

If you want to know more about how R works under the hood and the R core implementation, then I've provided a link on the slide there for a document about the internals, but I will emphasize that

I would guess that the majority of people using R are not coming from a traditional programming background and so -- so the interest in the core of R tends to be limited to a small group.

So a little bit about how the R community itself is structured, and so the R Foundation is a non-profit organization that supports the R community and also runs an annual Use R Conference held in different places around the world, usually on a rotating schedule, and so that people no matter where they are can attend easily, so this year it's going to be in Australia. Last year it was in Belgium. And then I think year after next, it's going to be in France, so and then I think it will be in the U.S. -- yeah, Boston the following year, so they try to rotate it around with Europe kind of being the regular place since that tends to be a more central place for a lot of the users.

And so they are a non-profit organization and they support -- they support the R Community through a lot of community-based initiatives, language-based initiatives, and it's kind of interesting, the R Consortium is the second foundation non-profit group that is part of the Linux Foundation, and so they do a lot of the same stuff as the R Foundation, except they're more focused on the community and enterprise interests, so they try to support the R Foundation, but they also try to provide that conduit as R becomes more used in the enterprise and you have different interests and organizations wanting to interact, the foundation can only do so much with regards to that, so the R Consortium as the Linux Foundation has all the same and premium membership means you get a position on the board of directors, the ISC, the internal steering committee which determines what proposals get funded and the marketing committee, so by proposals, anybody from the community can submit a proposal to the R Consortium, and that's pretty neat. You can get money from them if you need money for your effort. A lot of times, I mean, and if you don't, then you can also get recognition about your project and get other people engaged with it, so that's a nice thing that they offer.

And then the plot on the slide is the downloads of Top CRAN packages. CRAN is the package manager. There are a few different one, but CRAN is kind of the big one where anything you want, any kind of library or functionality you're going to get it from CRAN.

These are -- these are -- so I think I was just at R Studio (?) last week or the other week, I'm sorry, and it was mentioned that for these that have been specifically labeled, it's only about 40% of the CRAN packages are represented here, and so this isn't like by any means comprehensive, but it does give you a sense of what people are doing with R in terms of what they're downloading, and all of the CRAN data are available for free in a package, in an R package that you can download into R and make plots with, so that's pretty neat.

And then one thing that I think is really cool that's happened in R recently, I first used R in 2012 to do server capacity planning, and it was neat but it was -- I found it just not to be the most intuitive. Some of the workloads were a little funny, especially for me being used to Pearl, it just wasn't apparent how to do different things.

And in recent years, a developer who works for R Studio and is very active in the community and has written a lot of cool libraries, and he's been pushing more of a functional programming approach which is very task oriented and this is called the tidyr, and this is great way to streamline analysis and makes the code really readable, really easy to use, and so here I have an example of what he publishes as his -- or my version of his version of a data science workflow, and so the idea is that you gather the data somehow, usually it's unstructured. I believe Jenny Brian another big person in the community who does amazing talks and tutorials has one called rectangularizing data, and so it's the idea you're getting all of this loose data and you're trying to put it into a table to analyze it, and so these are some of the tools that you can use to do that.

We've got Read R, and R Vest. R Vest lets you get HTML and parse things out, I've used it quite heavily, you're literally scraping web pages and turning it into a data frame, a table that you can then do analysis on.

Read R will take in pretty much anything. Like if somebody sends you the output of a MySQL query in text, can you use Read R to read it into a frame in one line, it's amazing.

It's pretty cool. And then these are different steps on the next slide, validate, summarize, visualize. I want to highlight the ployer and there was an old library called plyr that they're building on top or kind of modifying the name of, and this is a tool that just has a lot of functions that lets you do things in mass to your data frame.

So an example in the code on the slide there, under summarize, is filter and so that does what it says. It's like you specify for a particular column what you want to filter on. And then there is -- my favorite that I use a lot is this one called Separate. You might read something in that has a string of text and then this lets you specify how you want to split that text up into columns in your frame.

And so that, again, you can use this all on one line, and usually, once you get used to it it's not too hard to read.

And then another one of my favorites is GG Plot and that's how I made all the plots in this slide, or in the slide deck. You take your data frame, you say what columns you want to do and you say how you want to visual size it, and bam there you go.

And then finally, when you want to publish the results of your

research, R has this really great library called knitr that lets you output or put the output into a variety so you ask use latex, PDF, make web pages with it which is something I've been doing a lot of recently, and then shiny is a neat package that lets you create a JavaScript application that you can either run on a phone or in a web browser and host it, and what's really neat about that is these are people that are researchers that don't necessarily come from a programming background, and then they're able to share their researches interactive web application, so this is pretty huge, and so it's a really cool merging of the software engineering world and the research world and I think that's something that really makes R stand out as an amazing community.

Another thing regarding R is, one of the big questions is how to integrate R into my production workflow, so here I'm calling out Plumber, which is a library that lets you make your R code into an API, and then it -- you run an API service, and so anything that needs it will go through the API service and tubing the R code and this is significant because R supports a lot of statistical library functions or has libraries for some statistical functions that aren't necessarily present in other places, or it may just be for time purposes, you want to deploy something and then you want to move it into something, you know, maybe that works better for whatever you have deployed, but in the interim, you can use this to at least have the R code working.

And so just to clarify, so you can integrate with Java, and then it also -- basically R serves as a socket server and supports binary requests. It also will decorate the existing R source code with special comments and that's how it generates the REST API for it.

Let's see. So the current status of R is -- these are just some different -- if you look around the web, one of the things that I would caution as you that a lot of the top languages out there, and this is coming from somebody now working in data science, is to always clarify your sources and understand who the audience is. If it's based on a survey, consider who the audience might be so, for instance, Katie is a primary developer audience, so you may not have as many researchers, and so the R may not be accurately represented because of that and so, you know, so other languages may be more represented and so these are just some different views of R's usage. It is pretty difficult to track because again many of the researchers and people using R aren't really engaged in tech and they're not always engaged in the larger communities because this idea of Open Source Ccommunity is still pretty new in that environment.

So it does extend beyond traditional technology software fields, and PyData and R have strong scientific use. It's usually coming from traditional engineers and software engineering

background, whereas R is typically used by statisticians and researchers and started maybe with Matlab and SPSS and other things and via the free aspect of the language and then this community support in particular are some of the things that draw them to it. So these are some plots from the O'Reilly Data Science Survey, the O'Reilly Data European Survey, and then in the slide note this is a Google Scholar plot and that I thought was a really cool article about popularity and the author shares how he created the plot, how he went through Google Scholar and used the citations to determine what languages were being used in 2016 for the research, and so it's pretty fascinating and pretty neat to see. It will be interesting to see what that looks like now two years later.

So one of the big things happening in the R Community as I alluded to is the integration of software engineering workflows and software engineering techniques in order to improve the reproducibility of research and to improve collaboration, so you have more people getting on GitHub and using GitHub, the idea that if you publish a paper, you should also have a GitHub Repo along with your paper so that people can actually download your analysis and run it making your data available. All of these things, and so people are starting to use continuous integration, they're starting to care more about the performance of R as a language, and so some of the key things, NumFOCUS is one of the big groups driving this, and the software carpentry program, the software carpentry and data program, they decided to make one program called the carpentries, so there is also an initiative to streamline the R user group experience to better unify the community. This year the R consortium is rolling out a meetup account to have all the R User Groups, register under that so people can get the support engagement that they need in order to be effective at reaching the community and growing the community.

And one of the things that I'm really excited about is that they've been taking advantage of their diverse demographic initiatives to improve inclusion with traditionally underrepresented groups, and so I run R-Ladies, it's another R consortium initiative and our forward -- R Forward is the R foundation initiative in the same vein and it's really R-Ladies in particular is woman-focused -- or I call them lady-focused user groups. Not exclusive. It's not like this or that only, but it's really more to meet the needs of that population and encourage them and encourage diversity.

And the recently at some of the conferences I've been to, I've developed this (?) called the bathroom line heuristic which is how long are the bathroom lines after the keynotes or something like that, and I'm proud to say that at the R Conferences the ladies' lines are comparable to the men's lines, so they appear to be working or they are getting a really good diverse community. So it's a

really great opportunity for people that are more tech focused to learn and engage a little bit.

So within IBM, so our --

>> Are you there? We don't hear you.

>> TODD MOORE: Augustina?

>> AUGUSTINA RAGWITZ: Hello? Can you hear me?

>> TODD MOORE: Yes. Go back a couple of minutes. We lost you for a few minutes.

>> AUGUSTINA RAGWITZ: Oh, okay. Was it what's next? Current status? Where did you lose me?

>> TODD MOORE: Actually, at the beginning of that slide you were on. Keep going forward. Right there.

>> AUGUSTINA RAGWITZ: Oh, okay. Perfect. Okay. Great. Yeah, so R -- so let's see. (Laughing). Let's see where I was. Yeah, R at IBM, actually a lot of people internally use R for their day-to-day research, and we started an internal virtual group to connect all the people using R, and it's really cool to talk to people and see what they're doing with it. Externally, we provide free online courses for learning R and data science through our CognitiveClasp AI, one of my favorite things about that is even if you're not taking a class they have a lab environment and can you actually run a hosted R studio environment, so if you're not comfortable or having some issues installing and getting things working locally on your computer, you can actually run it in this environment and there is a lot of packages and things available and you can always contact the CognitiveClass to get a package added to your environment. So a great way, low-investment way to get started using R and I highly recommend it.

As well our data science experience supports a full R studio instance and I believe that's what CognitiveClass AI is using but then you get access to it as a student, so to speak.

And then you can also integrate with Spark and a variety of other backends. What can be challenging about using R locally is when you're using huge amounts of data so that's when you really need the support of a data engineering team or you become the data engineering team. (Laughing).

And so something like data Science Experience is great because all the tools are integrated so you're not having to then deal with all of that. You can focus on doing your research and then you just read the docs on how to get things to connect and then you're good.

We also have example projects using R that are published, so you can see how to potentially using it for your own research as well as how to use R tools that we provide, but certainly you can use whatever you want. R Studio provides a lot of really great tools as well for -- for smaller scale things like publishing your knit output, rpubs and hosting shiny apps as well. There are a lot of options out there.

And so if you want to get started using R, I would recommend you visit CRAN, install R, and download R Studio, if you're not ready to install it locally, sign up to CognitiveClasp AI and you can use their environment and have a running R studio and poke around with it.

In addition, there are popular community resources like there is a package that's a really popular one, a command-line program that walks you through different things about R. There is an amazing book that I keep referencing back to regularly called R For Data Science, it's available for free or you can get the dead tree version through most places where you can buy books, and then one of the things that I found really useful, I took a statistics for R course on Corsara through Duke University and the one who runs it, her specialty is in statistics education and she has you do labs are notebooks and you learn how to structure a data science project in addition to refreshing your statistics that you may not have had for years and years as is the case for some of us, but I found that to be -- and you can audit for free and you don't have to pay for the Corsara course.

And then additionally, I linked at the bottom of the slide a couple of projects that we've provided through IBM where people are using R to do some fun things, so you can follow along with those to see what's possible.

And then really the thing to know about the R community, if you want to get engaged, is that most of their interaction is actually happening in person rather than in the virtual space, which is unusual for a tech community.

So I would recommend you really want to attend local meetups and go to conferences, and if you prefer connecting online, Twitter is probably one of the best places and can you use the hashtag Rstat, I often use hashtag Rladies as well for things in that environment, but if you go on Twitter and search for Rsatas, Rstudio is another resource, great account to follow, and the book, R For Data Science, he mentioned a few sources he follows for what's new in R.

And then one of the big things too as far as contributing to the community is that the R Consortium does review proposals regularly, so even if your proposal doesn't need funding, you can potentially partner up with another company, university, or community group or whatever and it's just a really great way to get attention to a project that you might be working on, whether it's a more social community-oriented project or it is somehow improving some code or writing a package for R.

And if your proposal is approved, then you form a working group and then you can potentially get more community support. Yeah, that's all I got and I'm happy to take any questions.

>> There is one question, just one that just came through. It says, are there any capabilities to use R with parallel computing

across nodes or machines?

>> AUGUSTINA RAGWITZ: Yeah, so there is a project called -- I should have it in the slides. Maybe I don't anymore. Maybe that was an old version. I believe, oh, yeah here we go. What's next for the R community. You might want to check out some of the high performance computing work that's being done as well as -- I'm trying to see. Oh, yeah the Big Data and Cloud improvements so I just list add couple of projects funded by the R Consortium, but distributed computing is definitely one of the things that is being worked on, and so I don't personally know a lot about that, but I would say probably the first -- or at least if I was going to find out more, is I would go to the R Consortium Proposals, I'll go back to that slide so it's up, but I think you all have access to the slides or have copies of them.

Yeah, I would check out who is doing those proposals, and then maybe reach out to them and see what the story is. As well, I feel like this is a Use R Conference you can look at Conference talk from the past years and see who has been presenting on it and what they've shared about it, so I mean unless somebody -- maybe somebody else on the call or somebody else has more information, you know, feel free to share in Slack or yeah.

>> That's it. No more.

>> Okay.

>> TODD MOORE: Thank you, Augustina. That was really good. I appreciate it, thank you for the questions, audience. This concludes Day 4 of our Open Source Week here at IBM and the week. Thank you all for coming and participating. These videos will be available for review as well as presentation material. Again, thank you, and my best wishes for a happy rest of the week.

That's it for today.

(completed at 1:54 p.m. CST)